EXHIBIT 7

IN THE UNITED STATES DISTRICT COURT
FOR THE EASTERN DISTRICT OF TEXAS
TYLER DIVISION

IMPLICIT, LLC,	§	
	§	
Plaintiff,	§	
	§	Civil Action No. 6:17-cv-336-JRG
vs.	§	(CONSOLIDATED CASE)
	§	
PALO ALTO NETWORKS, INC.,	§	
	§	
	§	
Defendant.	§	
	§	

AMENDED EXPERT REPORT OF DR. SAMUEL RUSS REGARDING THE INVALIDITY OF THE '683, AND '790 PATENTS

B. Prior Art

1. Decasper

77. The family of references referred to as the "Decasper references" or, collectively, "Decasper" are a set of articles written by Dan Decasper (*et al.*) in (or before) 1998. Two of these articles are: "Router Plugins - A Software Architecture for Next Generation Routers" (September, 1998) ("Decasper 1") (DEFS_IMPL_060988), and "DAN: Distributed Code Caching for Active Networks" (April 30, 1998) ("Decasper 5") (DEFS_IMPL_060952).

78. Decasper 1 was publicly available at least as early as September 4, 1998, when it was presented at the ACM SIGCOMM'98 conference, which took place from September 3-4, 1998. I am familiar with, and have personally attended, many conferences like the ACM SIGCOMM conference. In my experience, the papers presented at those conferences are provided to the attendees. My experience is consistent with the testimony Mr. Decasper gave, in which he explained that when he presented at the SIGCOMM'98 conference, the Decasper 1 paper was given out to attendees, and that he presented his paper to approximately a thousand attendees. (IMPL_GRP.B_00012616, May 17, 2017 Transcript of Decasper Deposition at 22-26.)

79. Decasper 5 was publicly available at least as early as April of 1998. I have attached as Exhibit C to my report the declaration of Ingrid Hsieh-Yee, who comfirmed public availability of Decasper 5 as of April 1998.

80. The references are treated as a single, unified disclosure inasmuch as they were written by the same person (Dan Decasper) in the same time period (1998) and describe the same ongoing research and development efforts.

81. I conducted a research program at Mississippi State University in the same time period (1994-1999) and, like Dr. Decasper, published a series of conference papers and journal

articles on the research. I regarded it as a single body of research, and the publications on the subject represented overlapping teachings.

82. To the extent the Decasper references are not considered a single, unified disclosure, it would have been obvious to a POSITA to combine their teachings. They stem from the same author, represent ongoing research work, are drawn to solve the same set of problems, and are drawn to the same field of art. There would have been a strong motivation to combine, inasmuch as the articles offer a series of overlapping solutions to the same problems, and would have been able to be combined without undue experimentation by a POSITA and would have been likely to yield predictable results.

83. Decasper 1 discloses the notion of router plugins. The plugins "allow code modules… to be dynamically added and configured at run time." The plugins can be bound to individual flows, and the article describes the packet classification needed to identify which plugins are to be used with each flow. (Note that the "flow" of Decasper is akin to the "message" of the Demux Patents.)

84. Decasper 1 also explains the process by which information from different headers is identified: "our filter table implementation targets only the Internet protocol stack, and requires packets to be classified based upon the same five packet header fields and the interface on which the packet was received. Our goal was therefore to find a fast lookup algorithm for matching the six-tuple <source address, destination address, protocol, source port, destination port, incoming interface> in a packet against a possibly large set of filters …" (Decasper 1 at p. 235.)

85. This system is illustrated in Fig. 3 of Decasper 1, shown below.

Networks by Andrew Tanenbaum ("Tanenbaum"), which was published in 1996. (DEFS_IMPL_061078.)

102. It would have been obvious to a POSITA to combine their teachings. The three RFC's define the TCP standard, and the textbook describes the TCP standard. A POSITA seeking to implement TCP (or to add TCP to an existing device) would have been motivated to consult all of these sources. There would have been a strong motivation to combine, inasmuch as they all describe the TCP standard in different ways, and would have been able to be combined without undue experimentation and would have been likely to yield predictable results.

VII. DETAILED OPINIONS RELATING TO THE INVALIDITY OF THE '683 PATENT

103. For the reasons detailed below, it is my opinion that claims 1, 2, and 4 of the '683 patent are invalid.

A. The Asserted Claims of the '683 Patent are Obvious in View of the Decasper Family of References

104. Claims 1, 2, and 4 of the '683 patent are invalid as being obvious in view of the Decasper family of references, collectively referred to as "Decasper".[1]

1. Claim 1 is Obvious in View of Decasper

> 1. A first apparatus for receiving data from a second apparatus, the first apparatus comprising:
> a processing unit; and
> a memory storing instructions executable by the processing unit to:
> create, based on an identification of information in a received packet of a message, a path that includes one or more data structures that indicate a sequence of routines for processing packets in the message;

[1] As noted above, the "Decasper references" or, collectively, "Decasper" are a set of articles written by Dan Decasper (*et al.*) in (or before) 1998. They include "Router Plugins - A Software Architecture for Next Generation Routers" (October, 1998) ("Decasper 1") and "DAN: Distributed Code Caching for Active Networks" (April 30, 1998) ("Decasper 5").

store the created path; and
process subsequent packets in the message using the sequence of routines indicated in the stored path, wherein the sequence includes a routine that is used to execute a Transmission Control Protocol (TCP) to convert one or more packets having a TCP format into a different format.

105. This claim is obvious in view of Decasper.

(i) "A first apparatus for receiving data from a second apparatus, the first apparatus comprising:"

106. If the preamble is limiting, Decasper discloses "[a] first apparatus for receiving data from a second apparatus".

107. The router of each Decasper reference is a "first apparatus" recited in the claim. The "second apparatus" is, for example, any device that sends the data to the first apparatus.

108. See, *e.g.*, Decasper 1 at Fig. 1 and Decasper 5 at Figure 2. The references also discuss packet processing by network hardware. "When a packet arrives, it gets passed to the IP core by the network hardware." (Decasper 1 at p. 233.)

(ii) "a processing unit; and"

109. Decasper discloses "a processing unit".

110. The router of each Decasper reference contains a "processing unit."

111. For example, one set of software disclosed in Decasper ran on a 233 MHz Pentium. (Decasper 1, p. 229 (Abstract)). "The hardware architecture for the proposed Active Network Node (ANN) is shown in Figure 4. It is derived from our high performance IP routing architecture [20] and refined and optimized for the purpose of active networks. The node consists of a set of Active Network Processing Elements (ANPE, four in Figure 4) connected to an ATM switch fabric [9]." (Decasper 5, p. 614.)

Plaintiff's logic, at least one place where Decasper discloses "applying one or more routines, where at least one such routine is a conversion routine" is in connection with the performance of TCP.

121. Moreover, to the extent that a claimed "conversion routine" in the context of the Demux Patents is a routine that converts a packet from a format suitable for one network protocol layer to another, Decasper 1 and 5 disclose "conversion routines" and/or "packet conversion." For example, as discussed further below in connection with '683 Patent claim 1(vii), the TCP protocol running on the router of Decasper 1 must be converting the packets to IP format for transmission. Further, the packets have to be converted from IP format to a native transmission format, such as *e.g.* ATM for transmission. (Decasper 1 was tested using ATM. See Decasper 1 at p. 239.) Therefore, Decasper discloses "a sequence of routines for processing packets in the message."

122. Decasper 1 discloses an architecture in which packets are classified into "flows." Summarizing pp. 232-233 of Decasper 1, the system has an IPv4/IPv6 core with "gates." Each gate is a point at which optional packet processing can occur. The system also has an "Association Identification Unit" (AIU). If a packet arrives that belongs to an uncached flow (belongs to a flow that has not yet been identified), the packet is looked up using a "filter table" that is used to identify a "plugin instance." Each entry in the flow table contains pointers to the plugins, which are the routines required for processing the packets associated with that specific flow.

123. Decasper 5 discloses a system in which network nodes (such as routers) can fetch the code needed for packet processing from a remote location at runtime. Decasper 5 describes a

network-packet format in terms of a series of "function identifiers," which are cues found in each network-layer header. This is illustrated in Figure 1 of Decasper 5:

Figure 1: Datagram (schematically)

(Figure 1 From Decasper 5.)

124. In an example found in Decasper 5, an Ethernet packet contains an identifier for IPv4 or IPv6. (Decasper 5 at p. 610.) This is named a "function identifier" because it contains the information needed to determine which processing steps the packet requires. If a network node encounters a function identifier it does not know how to process, the node "temporarly suspends processing of the packet and calls a 'code server' for the implementation of the function" that is needed to process the packet dynamically at runtime. (Decasper 5 at p. 611.) Decasper refers to such a node as an Active Network Node or ANN. This process is also shown in Fig. 2 of Decasper 5.

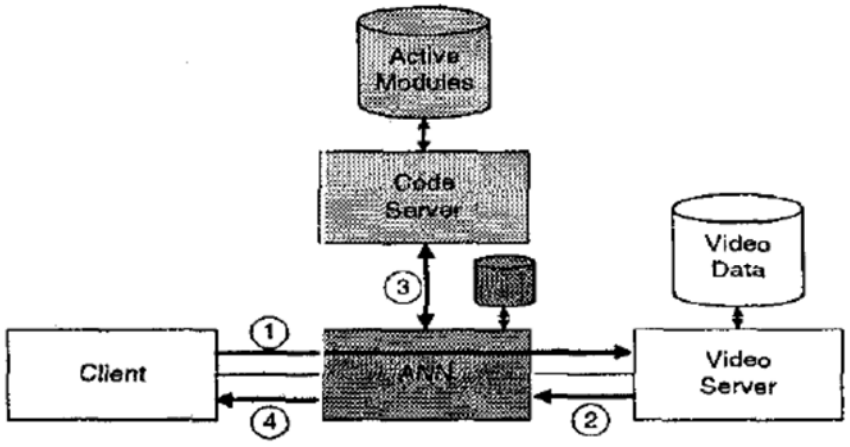

Figure 2: ANN fetching active module from code server

(Figure 2 from Decasper 5.)

125. Decasper 1 further explains: "[a] note on our use of the word 'plugin' (instead of 'module') is in order. In the web browser world, a plugin is a software module that is dynamically linked with the browser and is responsible for processing certain types of application streams (or flows). In a similar fashion, our router plugins are kernel software modules that are dynamically loaded into the kernel and are responsible for performing certain specific functions on specified network flows." (Decasper 1, p. 230.)

126. "The primary goal of our proposed architecture was to build a modular and extensible networking subsystem that supported the concept of flows, and the ability to select implementations of components based upon flows (in addition to simple static configurations). Because the deployment of multimedia data sources and applications (e.g. real-time audio/video) will produce longer lived packet streams with more packets per session than is common in today's environment, an integrated services router architecture should support the notion of flows and build upon it. In particular, the locality properties of flows should be effectively exploited to provide for a highly efficient data path." (Decasper 1, p. 229.)

127. "The AIU is responsible for maintaining the binding between flows and plugin instances. It makes use of a special data structure called a flow table to cache flows. Flow tables allow for very fast lookup times for arriving packets that belong to cached flows… If an incoming packet belongs to an uncached flow, its lookup in the flow table data structure will fail (i.e., there is a cache miss). In this case, the packet needs to be looked up in a different data structure that we call a filter table… The filter table lookup algorithm finds the most specific matching filter (described later) that has been installed in the table, and returns the corresponding plugin instance… Each flow table entry stores pointers to the appropriate plugins for all gates that can be encountered by packets belonging to the corresponding flow. The processing of the

130. Decasper 5 discloses the following: "[f]igure 4 shows an example data flow coming into the ANN at ANPE A going out at ANPE D. The active processing is done in ANPE C since ANPE A is heavily loaded and the load-sharing algorithm directed the flow to ANPE C which finally directs the flow to the ANN connected to ANPE D (ANPE A and D switch the flow in hardware without CPU intervention through the APIC)." (Decasper 5, p. 614.)

(v) "store the created path; and"

131. Decasper discloses "stor[ing] the created path."

132. Decasper 1, for example, discloses that each flow has a "flow table entry" that stores pointers to the functions needed to process packets. (See the discussion '683 Patent Claim 1(iv) *supra* for more details.)

133. Decasper 5, for example, also discloses that the fetched processing code for the ANN is stored on the node for use with subsequent packets in the flow: "[o]nce the module is downloaded, it is permanently stored locally on the ANN preventing other downloads for the same active module in the future." (Decasper 5 at p. 611.)

(vi) "process subsequent packets in the message using the sequence of routines indicated in the stored path,"

134. Decasper discloses "process[ing] subsequent packets in the message using the sequence of routines indicated in the stored path".

135. Decasper 1 uses the flow table to process subsequent packets using the same sequence of routines as used for the first packet as refenced in '683 Claim 1(iv) above. Once the flow table has been constructed, the packet, and subsequent packets in the same flow, can be processed by using the pointers in the flow table to invoke specific packet-processing routines. Subsequent packets can use the flow table directly, without having to go through the process of

flow table construction. In the language of Decasper, these are called "cached flows." Decasper 5 uses the downloaded code at the ANN to process subsequent packets in a flow.

136. Decasper discloses the following: "[a]n entry for a flow in the flow table serves as a fast cache for future lookups of packets belonging to that flow. Each flow table entry stores pointers to the appropriate plugins for all gates that can be encountered by packets belonging to the corresponding flow." (Decasper 1 at p. 233.)

137. "Once the functions are loaded by the node, they are in no way different than the ones compiled into the network subsystem at build-time. For example, the functions have as much control over the network subsystem's data structures as any other function in the same context, they are executed as fast as any other code." (Decasper 5 at p. 611.)

(vii) "wherein the sequence includes a routine that is used to execute a Transmission Control Protocol (TCP) to convert one or more packets having a TCP format into a different format."

138. Decasper discloses "wherein the sequence includes a routine that is used to execute a Transmission Control Protocol (TCP) to convert one or more packets having a TCP format into a different format."

139. As noted elsewhere in this report, the Plaintiff appears to interpret the claim such that the mere performance of the well-known Transmission Control Protocol satisfies this claim step. To the extent that a claimed "conversion routine" in the context of the Demux Patents is a routine that converts a packet from a format suitable for one network protocol layer to another, Decasper 1 and 5 disclose "conversion routines" and/or "packet conversion." Note that this interpretation of conversion is assumed for the purposes of this analysis, as it appears to be the interpretation used by Plaintiff.

140. Decasper 1 indicates that the Transmission Control Protocol is being performed. For example, it discloses that an important capability of routers is "level 4 switching," which